M467 Secure ISP with Key Store

NuMicro® 32位系列微控制器范例代码介绍

文件信息

应用简述	本范例程序为 M460_ISP_HID 之进阶应用，M467 透过 ISP Programming tool 下载加密固件后，使用 Key Store 搭配 AES crypto 还原固件固件内容并进行更新，藉此完成安全下载。
BSP 版本	M460 Series BSP CMSIS V3.00.002
开发平台	NuMaker-M467HJ V1.0

The information described in this document is the exclusive intellectual property of Nuvoton Technology Corporation and shall not be reproduced without permission from Nuvoton.

Nuvoton is providing this document only for reference purposes of NuMicro microcontroller and microprocessor based system design. Nuvoton assumes no responsibility for errors or omissions.

All data and specifications are subject to change without notice.

For additional information or questions, please contact: Nuvoton Technology Corporation.

www.nuvoton.com

1. 概述

本范例程序提供用户使用 USB_HID 接口进行固件更新时，可先透过新唐的 CryptoTool 进行固件加密，再透过新唐 NuMicro_ISP_Programming_Tool 将加密后的固件写入待更新的 NuMaker-M467HJ 开发版。M467 在确认接收完成后会調用 XOM 内部的解密 API，搭配硬件加密加速器，以及储存在 KeyStore Flash 的 AES 密钥将加密固件进行解密，而后将固件写入 APROM 完成更新流程。

1.1 AES 加密

固件加密采用 Advanced Encryption Standard (AES) 进行，为一种对称式加解密算法，用作加解密的密钥为同一把，长度可为 128、192 或者 256 bits。方式为将要加密的数据分为数个固定大小的区块进行加密，其优点为安全性高、速度快，为美国国家标准与技术研究院(NIST)所发布。

目前 NIST 针对 AES 算法提供了多种模式，新唐 CryptoTool 采用 AES-CFB-256 ，在进行档案加解密前须设定 256 bits 的密钥，以及 128 bits 的 Initialization Vector (IV)，并指定需做加密档案的位置，以及加密完档案的存放路径及文件名，可参考如下:

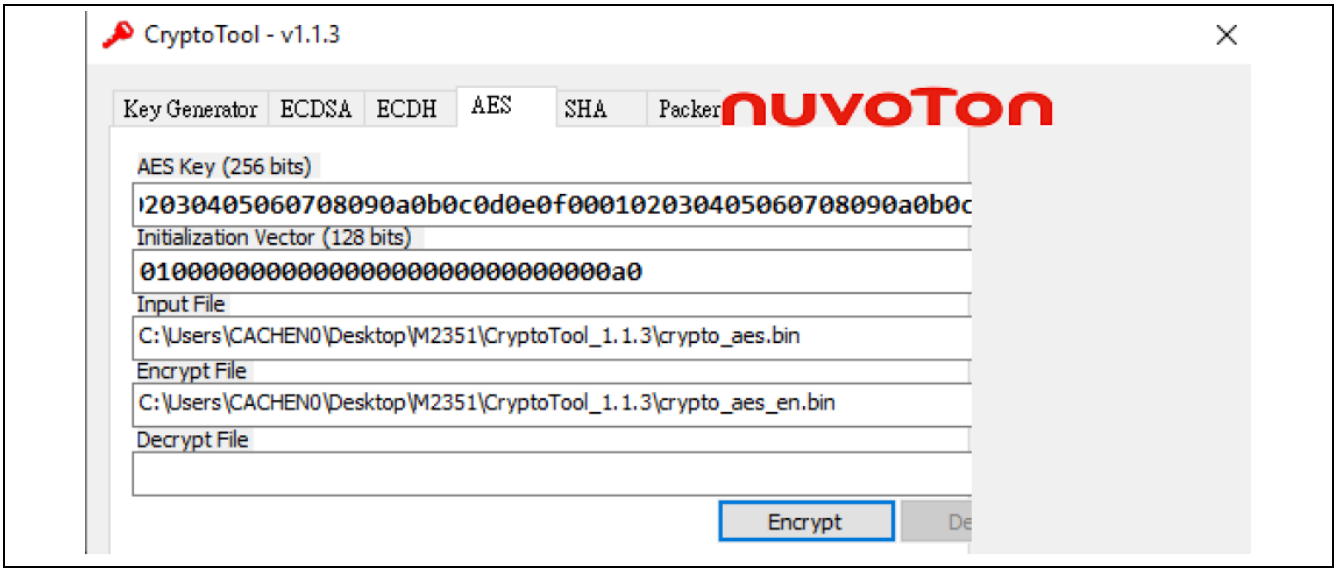


图 1-1 Nuvoton CryptoTool -v1.1.3

1.2 Execution Only Memory (XOM)

M467 可将内部 APROM 配置成独立的 XOM 区域，该区域储存的指令(程序代码)无法被读取只能执行，最多可配置 4 块独立的 XOM，用户可将机密程序代码放在该区域，防止外部程序意外读取数据，以及泄漏客户软件信息，详细说明可参考 M463/M467 Series Technical Reference Manual 章节 6.5.4.6。范例中我们将解密固件所用到的 API 以及 AES 加解密所使用的 IV 放在该区域内。

1.3 Key Store

M467 Key Store 作为密钥管理装置，提供安全存储的独立内存空间，可将密钥或重要数据存在 SRAM, Flash 以及 OTP，在进行密钥存储设定的过程中，可让该密钥只能用在特定硬件加密功能，举例来说，若设定存在 Flash 的 Key 1 是给 AES 进行使用，则 CPU 以及其他如 ECC 或者 RSA 的硬件加密就无法读取到该密钥，让密钥的保护达到更为安全及完善。详细的安全功能及使用说明请参考 M463/M467 Series Technical Reference Manual 章节 6.42。

1.4 软件架构

1.4.1 Key Store test

软件流程共分两部分来说明，第一部份 key store 写入测试如图 1-2 所示， 使用 NuMaker-M467HJ ，连接板上 Nu-Link2-me 及测试电脑后，按下 reset 键范例程序便开始运行。一开始会先初始化 Key Store，之后便会询问是否要进行清除 Key Store Flash 的动作，若开发板为第一次使用则不需进行清除，且会看到串口调试助手上显示” KS FLASH remained size = 4060”。串口调试助手下一步询问是否要进行 Key 写入至 Flash，若第一次使用则必须进行此动作，完成后会将已写入至 Key Store Flash 的密钥进行 AES CFB 模式的加解密测试，并将结果显示在串口调试助手上。

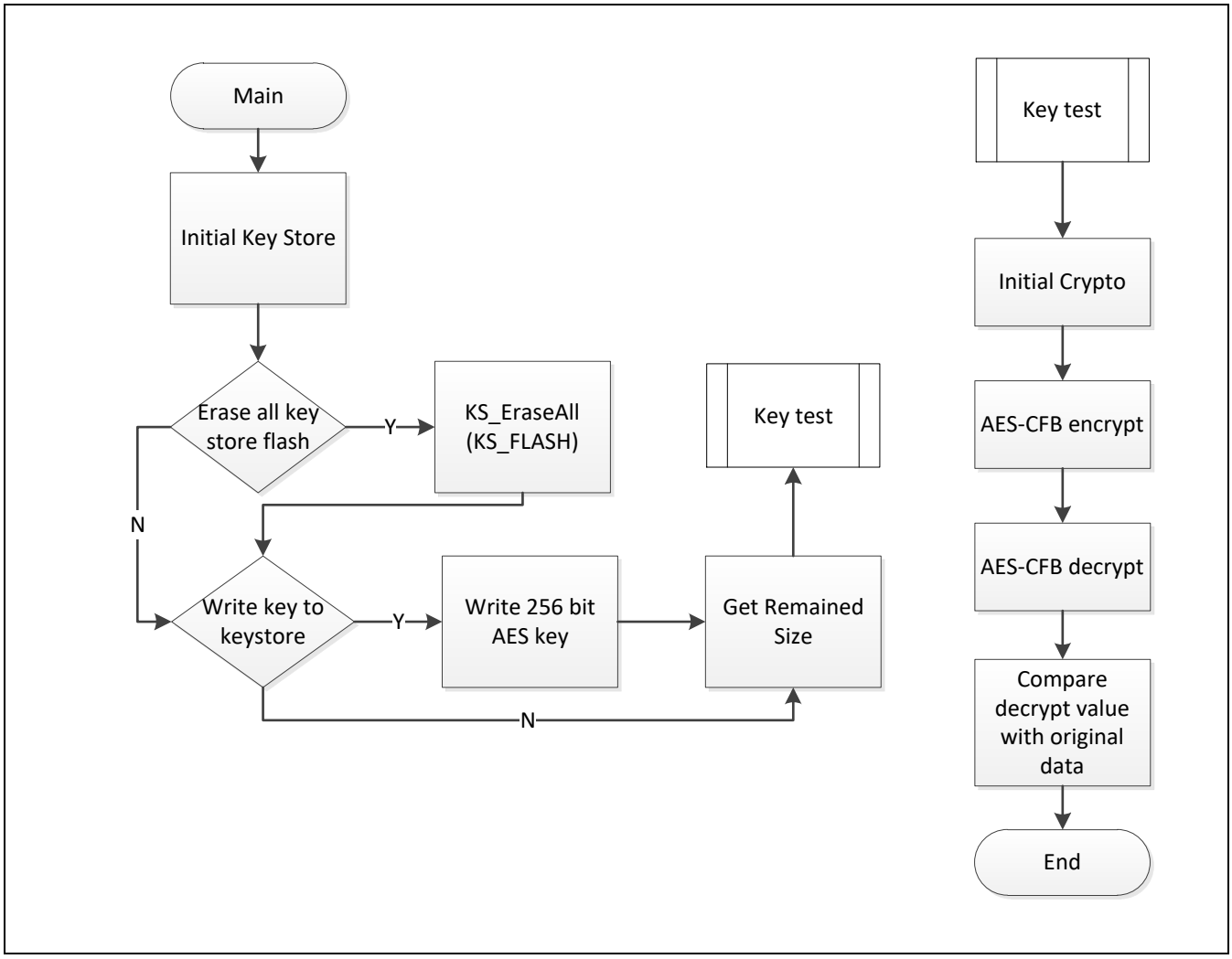


图1-2 Key test 软件流程图

1.4.2 Secure ISP 更新流程

第二部份为主要更新流程，共分两部分来说明，第一部份先初始化 USB 以及开启 ISP 功能，之后侦测 PC.13 的 IO 电平来判断是否进入 ISP 更新程序，若 PC.13 为 Low 则开始下载更新程序，为 high 则跳转到 APROM 执行。当下载更新完成后便会进到第二部分的 XOM 区域，执行解密以及写入 APROM 的动作，首先初始化加密引擎，接着设定相关密钥, 初始向量以及 DMA 地址后，加密引擎便会开始还原待更新的程序代码，最后将其写入 APROM。

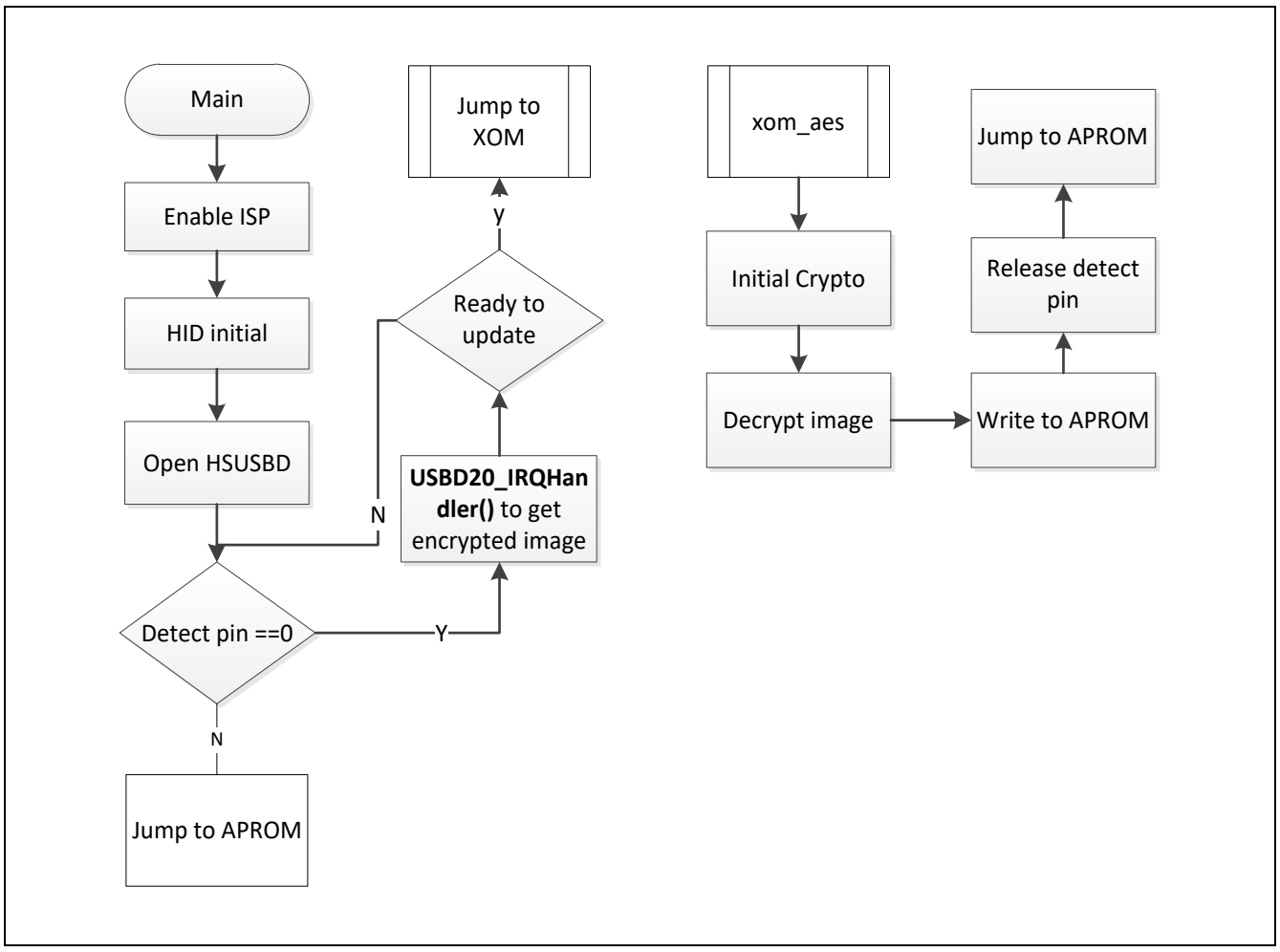

Main
Enable ISP
HID initial
Open HSUSBD
Detect pin ==0
N
Jump to APROM
Y
USBD20_IRQHandler() to get encrypted image
Ready to update
N
y
Jump to XOM
xom_aes
Initial Crypto
Decrypt image
Write to APROM
Release detect pin
Jump to APROM

图 1-3 软件流程图

1.5 功能介绍

1. 将 NuMaker-M467HJ 连接 PC 后，开启 Keil M467_KS_KeyTest project 范例，完成编译后下载至开发版上，执行 AES key 的写入测试。

```
KS FLASH remaind size = 4060
Erase all keystore flash?(y/n)
KS FLASH remaind size = 4060
write key to KeyStore?(y/n)
KS FLASH remind size: 4024
The key in key store:
key number is: 1
0x0000  03 02 01 00 07 06 05 04 0b 0a 09 08 0f 0e 0d 0c   ................
0x0010  03 02 01 00 07 06 05 04 0b 0a 09 08 0e 0e 0d 0c   ................

The input data:
0x0000  00 11 22 33 44 55 66 77 88 99 aa bb cc dd ee ff   .."3DUfw........
0x0010  00 11 22 33 44 55 66 77 88 99 aa bb cc dd ee ff   .."3DUfw........

AES encrypt done. The output data:
0x0000  ed 62 a0 fe f1 da e1 41 cc a7 ba 2b 91 9d 19 49   .b.....A...+...I
0x0010  a1 77 bd 71 7b 7f e1 53 65 bf f7 64 11 7e d1 1e   .w.q{..Se..d.~..

AES decrypt done. Decode Data:
0x0000  00 11 22 33 44 55 66 77 88 99 aa bb cc dd ee ff   .."3DUfw........
0x0010  00 11 22 33 44 55 66 77 88 99 aa bb cc dd ee ff   .."3DUfw........

Done!
```

图 1-4 AES key 测试流程

2. 透过 ISP Programming Tool 下载加密过后的 image，XOM 内的程序代码会将 image 解密后写入至 APROM 并跳转到 APROM 执行。

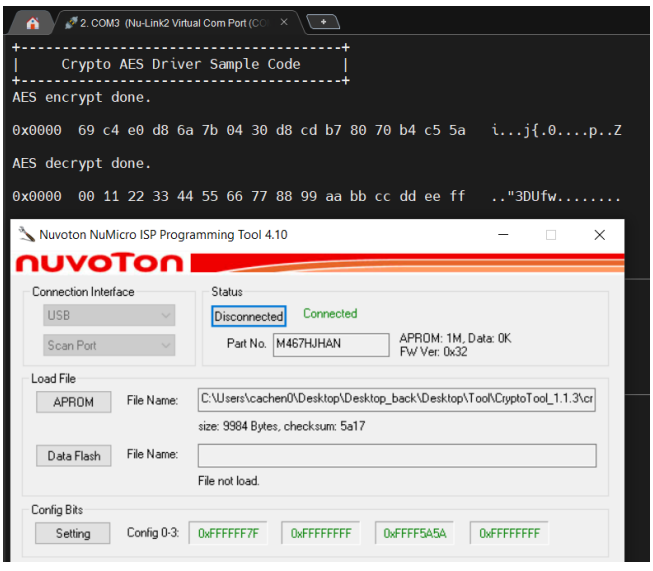


图 1-5 ISP Programming Tool 更新结果

2. 程序介绍

2.1 Key Store test

以下程序实现了章节 1.4.1 的架构，核心是要实现密钥的写入及功能测试，其中关键的 AES key 透过 au32Key 数组实现，用户在孰悉操作后可自由替换成实际产品所使用的密钥，并妥善保存。AES 密钥属性，建议设定为 CPU 不可读取，这样可确保该密钥只能被 AES engine 所取用，藉此达到完整的保护。AES 密钥在成功写入后会返回一编号(i32KeyIdx)，用户在調用 AES engine 搭配 Key Store 做加解密时，需填入正确的编号才能成功还原原始数据。

```
int main(void)
{
    int8_t i8Test = 0;
    int32_t i32KeyIdx = 1;
    /* au32Key value is written into Key Store Flash which is used to decrypt image.
    au32Key value is determined by the user and should be the same with CryptoTool. */
uint32_t au32Key[8] = {
        0x00010203, 0x04050607, 0x08090a0b, 0x0c0d0e0f,
        0x00010203, 0x04050607, 0x08090a0b, 0x0c0d0e0f
    };
    char     chr;

    /* Unlock protected registers */
    SYS_UnlockReg();
    /* Init System, peripheral clock and multi-function I/O */
    SYS_Init();
    /* Init UART for printf */
    UART_Init();
    /* Init Key Store  */
    KS_Open();

    printf("KS FLASH remained size = %d\n", KS_GetRemainSize(KS_FLASH));

    printf("Erase all keystore flash?(y/n)\n");
    chr = getchar();
    if((chr == 'y') || (chr == 'Y')) {
        KS_EraseAll(KS_FLASH);
        printf("KS FLASH remaind size = %d\n", KS_GetRemainSize(KS_FLASH));
    }
    printf("write key to KeyStore?(y/n)\n");
    chr = getchar();
    /*---------------------------------------
    *  Key attribute: flash key
    *  Key meta data: AES-256 bits
    *  Return Value :
    *  i32KeyIdx, the index value should be set
    *  when using crypto encrypt/decrypt with
    *  key store flash key
    *---------------------------------------*/
    if((chr == 'y') || (chr == 'Y')) {
        i32KeyIdx = KS_Write(KS_FLASH, KS_META_256| KS_META_AES, au32Key);
    }
    if(i32KeyIdx < 0) {
        printf("Fail to write key to Key Store!\n");
        printf("KS FLASH remaind size = %d\n", KS_GetRemainSize(KS_FLASH));
        return -1;
    }
```

```
    printf("KS FLASH remind size: %d\n", KS_GetRemainSize(KS_FLASH));

    printf("The key in key store:\n");

    printf("key number is: %d\n", i32KeyIdx);
    DumpBuff((uint8_t *)&au32Key[0], 32);
    /* use AES_Test to check the encryption and decryption with Key Store */
    i8Test = AES_Test(CRPT, KS_FLASH, i32KeyIdx);
    if(i8Test == 0) {
        printf("Done!\n");
    } else {
        printf("fail!\n");
    }
    while(1) {}
}
```

2.2 ISP_HID_20_SecureUpdate

以下程序实现了章节 1.4.2 的架构，前半段透过 USB 以及 ISP 协议，将加密的固件写入到事先规划好的 SRAM 区域(参考 *fw_image.c* 以及 *scatter.sct*)，后半部在 void UpdateRoutine(void) 实现了安全更新的细节，透过 XOM 进行保护，让黑客无法得知详细流程指令。Secure ISP 程序提供给客户一个安全更新的基本范例，客户可根据其具体需求，改写 UpdateRoutine 的流程，如更动使用的 KEYNUM，或者调整 AES 加解密模式以及 IV 值，藉此来完善属于产品本身的更新机制。

```
int32_t main(void)
{
    /* Unlock write-protected registers */
    SYS_UnlockReg();

    /* Init System, peripheral clock and multi-function I/O */
    if(SYS_Init() < 0) {
        goto _APROM;
    }

    /* Enable ISP */
    CLK->AHBCLK0 |= CLK_AHBCLK0_ISPCKEN_Msk;

    FMC->ISPCTL |= FMC_ISPCTL_ISPEN_Msk | FMC_ISPCTL_APUEN_Msk | FMC_ISPCTL_ISPFF_Msk;

    /* Get APROM and Data Flash size */
    g_u32ApromSize = GetApromSize();
    GetDataFlashInfo(&g_u32DataFlashAddr, &g_u32DataFlashSize);
    if(DetectPin != 0) {
        goto _APROM;
    }

    /* Open HSUSBD controller */
    HSUSBD_Open(NULL, NULL, NULL);

    /* Endpoint configuration */
    HID_Init();

    /* Start transaction */
    HSUSBD->OPER = HSUSBD_OPER_HISPDEN_Msk;    /* high-speed */
    HSUSBD_CLR_SE0();
```

```
    g_u8ISPDone = FALSE;
    /* check PC.13 input status */
   while(DetectPin == 0) {
        /* Polling HSUSBD interrupt flag */
        USBD20_IRQHandler();

        if(g_u8UsbDataReady == TRUE) {
            ParseCmd((uint8_t *)g_u8UsbRcvBuff, 64);
            EPA_Handler();
            g_u8UsbDataReady = FALSE;
        }
        if(ISPDone == TRUE) {
            UpdateRoutine();
            g_u8ISPDone = FALSE;
        }
    }

_APROM:
    /* Reset system and boot from APROM */
    /*set vector to APROM "with IAP", only use with IAP need to set vector */
    FMC->ISPCMD = FMC_ISPCMD_VECMAP;              /* ISP command */
    FMC->ISPADDR = 0x00000000;                    /* Vector remap address */
    FMC->ISPTRG = FMC_ISPTRG_ISPGO_Msk;           /* Trigger ISP command */
    while(FMC->ISPTRG & FMC_ISPTRG_ISPGO_Msk) ;   /* Wait for ISP command done. */
    /*mcu reset and start from APROM*/
    SYS->IPRST0 |= SYS_IPRST0_CPURST_Msk;
    /* Trap the CPU */
    while(1);
}
    /* Secure fw update procedure */
void UpdateRoutine(void)
{

    uint32_t StartAddress = 0x0000,  TotalLen;
    TotalLen = g_au32Encryptedimage[0];

    /*---------------------------------------
    *  Step 1, decrypt image
    *  AES-256 CFB mode decrypt fw image
    *---------------------------------------*/
    SYS_UnlockReg();
    KS_Open();
    AES_Open(CRPT, 0, 0, AES_MODE_CFB, AES_KEY_SIZE_256, AES_IN_OUT_SWAP);
    /* Use key in key store */
    AES_SetKey_KS(CRPT, KS_FLASH, KEYNUM);
    AES_SetInitVect(CRPT, 0, (uint32_t*)au32MyAESIV);
    AES_SetDMATransfer(CRPT, 0, (uint32_t)&g_au32Encryptedimage[1],
(uint32_t)au8OutputData, TotalLen);
    AES_Start(CRPT, 0, CRYPTO_DMA_ONE_SHOT);
    while(((CRPT)->AES_STS & CRPT_AES_STS_BUSY_Msk));

    /* Step 2, update APROM */
    FMC_ENABLE_AP_UPDATE();
    WriteData(StartAddress, StartAddress + TotalLen, (uint32_t *)au8OutputData);

    return;
}
```

3. 软硬件需求

3.1 软件需求

- BSP 版本
  - M460 Series BSP CMSIS V3.00.002
- IDE 版本
  - Keil uVersion 5.39

3.2 硬件需求

- 电路组件
  - NuMaker-M467HJ V1.0
- 线路示意图
  - 杜邦线*1& 连接脚位(PC.13)

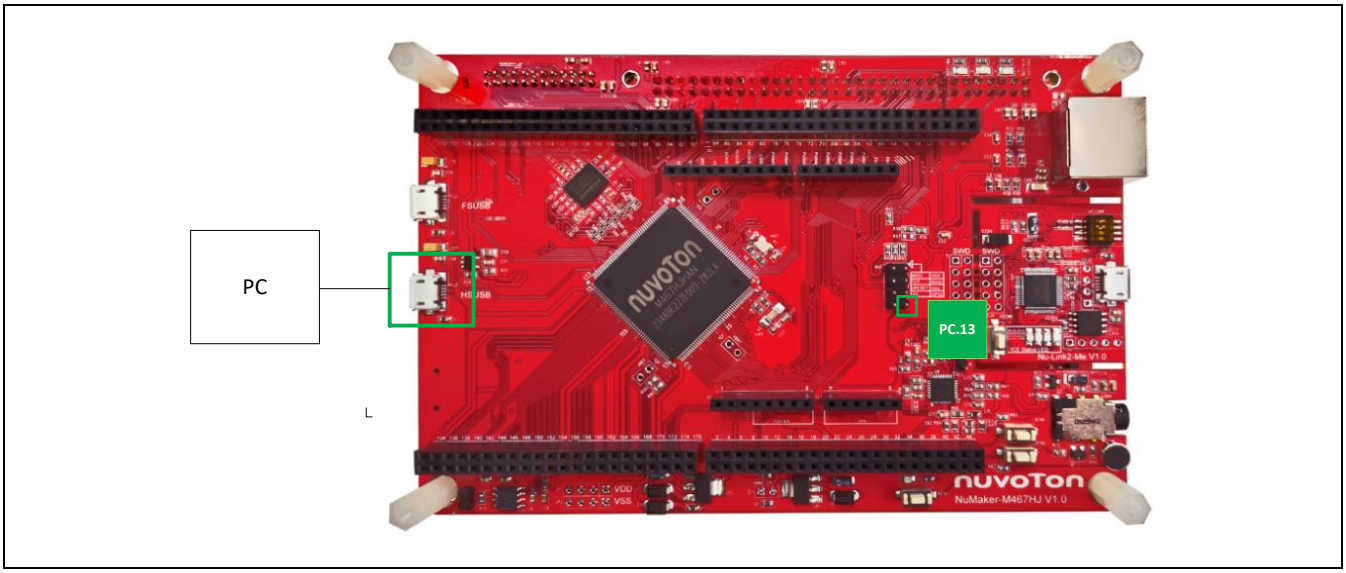


Figure 3-1 線路示意圖

4. 目录信息

目录架构如下所示。

EC_M467_Secure_ISP_with_Key_Store_V1.00	
Library	Sample code header and source files
CMSIS	Cortex® Microcontroller Software Interface Standard (CMSIS) by ARM® Corp.
Device	CMSIS compliant device header file
StdDriver	All peripheral driver header and source files
SampleCode	
ExampleCode	
Project	Source file of example code

图 4-1 目录信息

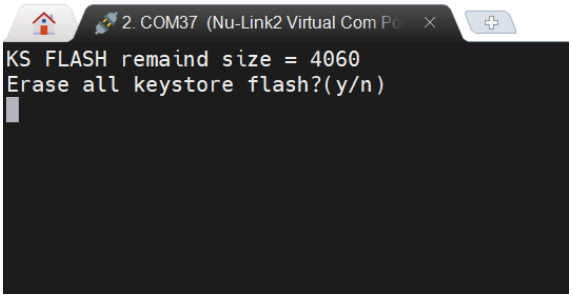
5. 范例程序执行

5.1 测试流程

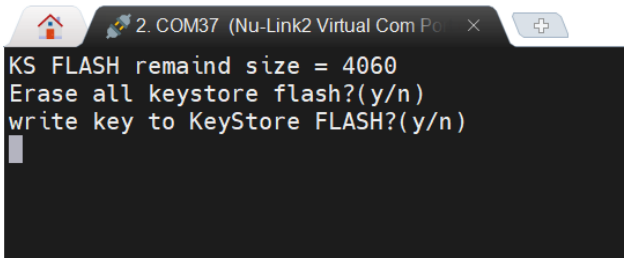
5.1.1 Key 设定及测试

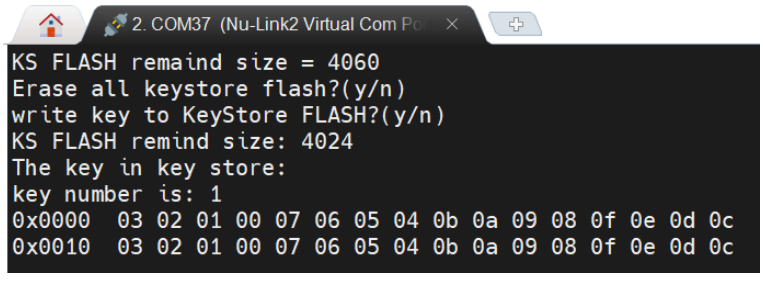
此范例将用于解密更新程序代码的 AES 密钥写入 Key Store 的 Flash 区域中，并设定为 AES 硬件加密专用，藉此保护密钥。

1. 打开 M467_KS_KeyTest 的 KEIL 文件夹，双击 *M467_KS_KeyTest.uvprojx*。
2. 编译范例程序后烧入至 NuMaker- M467HJ 并打开串口调试助手装置显示打印讯息。
3. 一开始显示 Key Store Flash 剩余空间，若为第一次使用则为 4060 bytes，不需做 Erase all key store flash 的操作。



4. 下一步会询问是否要进行 KeyStore 写 key 的动作，输入”y”或”Y”后会将范例程序内 au32Key 的数据写入，成功写入后会回传存放位置的编号，以及打印 au32Key 数据。





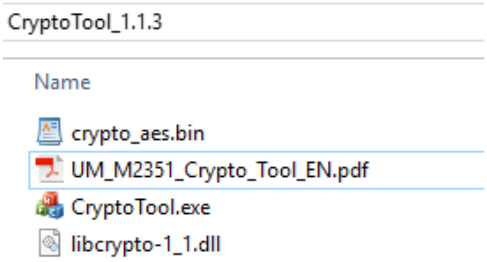
5. Key Store 写入完成后开始搭配 Crypto engine 进行 AES-CFB 测试，进行一次加密以及解密操作，比较完成后的数据与原始数据是否一致。

```
The input data:
0x0000  00 11 22 33 44 55 66 77 88 99 aa bb cc dd ee ff
0x0010  00 11 22 33 44 55 66 77 88 99 aa bb cc dd ee ff

AES encrypt done. The output data:
0x0000  09 c4 84 cf 9f da 7b e7 6b 00 71 2a c8 e3 78 bd
0x0010  0c 38 3d e8 5b 48 8b e9 18 43 62 e6 00 14 63 1c

AES decrypt done. Decode Data:
0x0000  00 11 22 33 44 55 66 77 88 99 aa bb cc dd ee ff
0x0010  00 11 22 33 44 55 66 77 88 99 aa bb cc dd ee ff

Done!
```

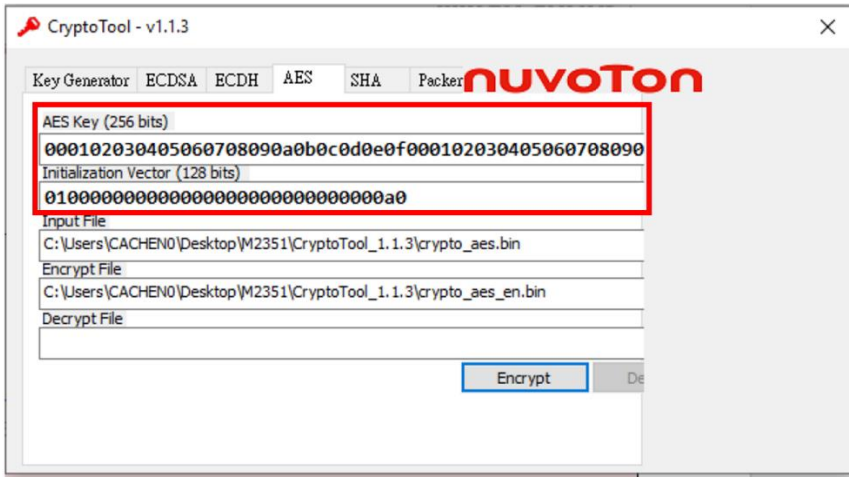
5.1.2 Crypto tool 加密更新档

新唐 Crypto tool 可至以下连结下载: [Nuvoton NuTool Crypto Tool]

1. 下载后执行 CrptoTool.exe。

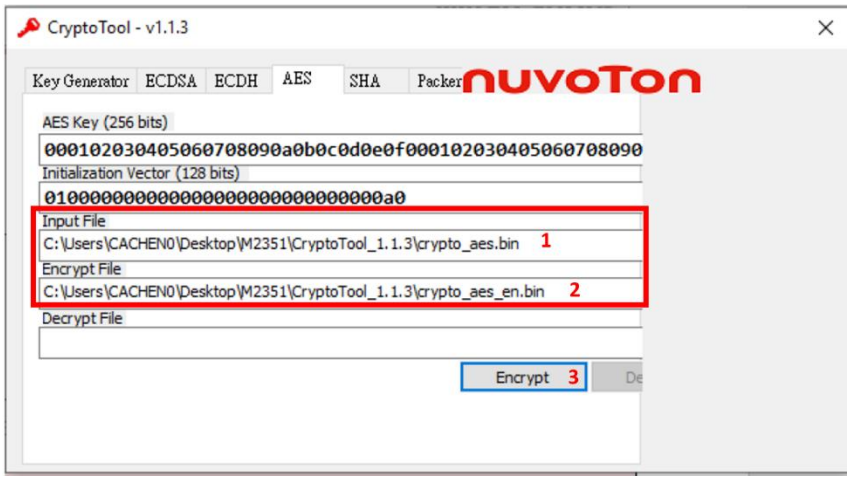


2. 选择 AES 字段，设定 AES key 以及 Initialization Vector。



3. 编译 M460 BSP 内任意一范例程序，并将编译完成的 bin 档复制到 CryptoTool 底下。

4. 设定 Input File 及 Encrypt File 的路径，按下 Encrypt 按钮，几秒后会在该路径下看到生成的加密 bin 檔。



5. 完成

5.1.3 ISP 程序下载及 XOM 设定

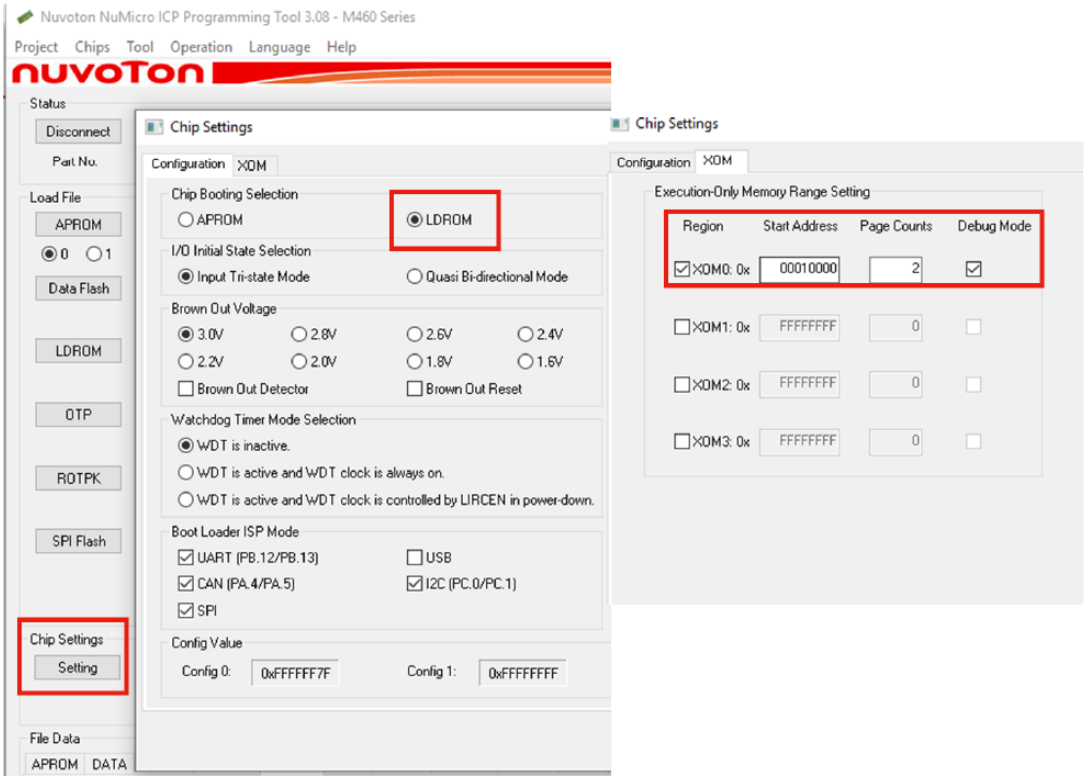
1. 根据目录信息章节进入 ExampleCode 路径中 M467_ISP_HID_20_SecureUpdate 的 KEIL 文件夹，双击 *M467_ISP_HID_20_SecureUpdate.uvprojx*。

2. 编译后按下 Download 写入至开发版。

3. 打开 Nuvoton NuMicro_ICP_Programming_Tool 点选 M460 series 连接开发版。

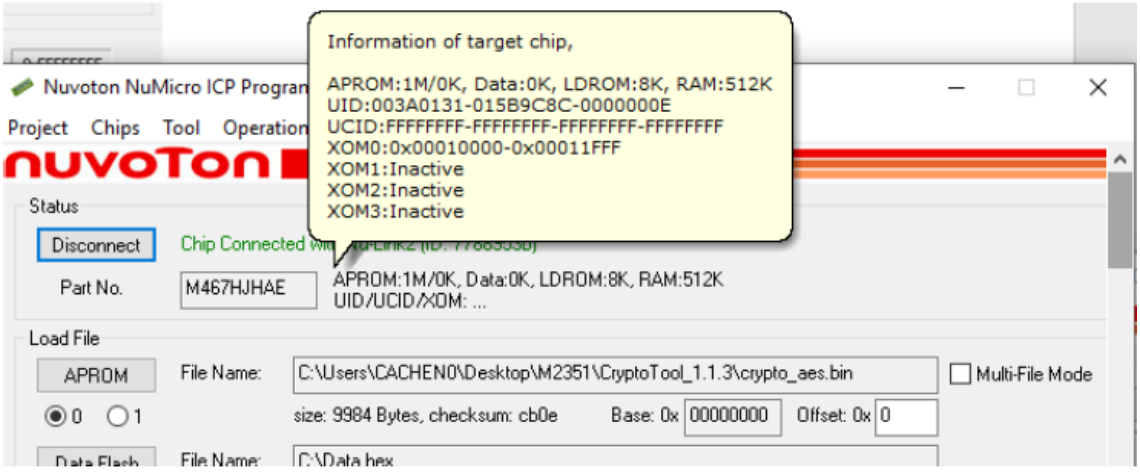
4. 点选 Chip Settings -> Chip Booting Selections 设定 LDROM。

5. 点选 Chip Settings -> XOM 依照下图红框设定 XOM，Debug Mode 不勾选则在 debug 模式下无法执行单步除错且看不到指令内容。



6. 确认 Programing 区域只勾选 Chip Setting 后按下 Start。

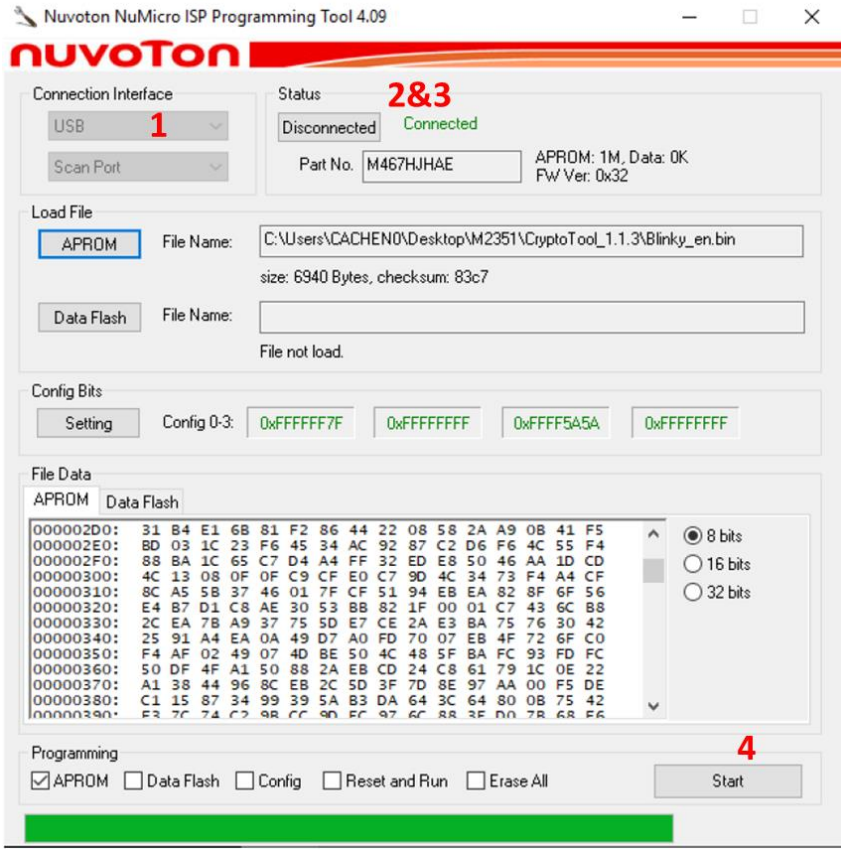
7. 重新连上 Tool，确认 XOM0 启动。



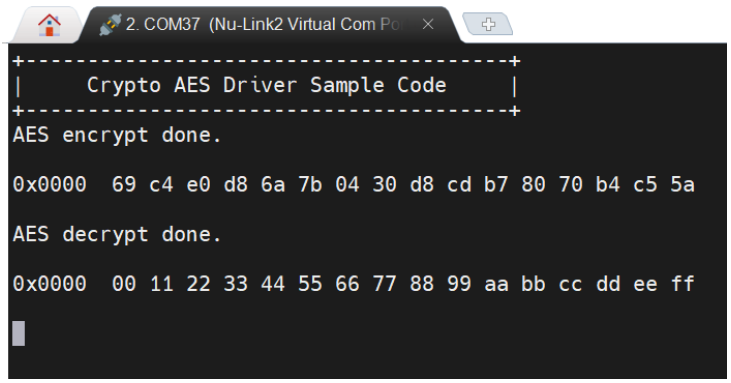
5.1.4 ISP 安全范例更新步骤执行

完成上述步骤后，本小节开始说明如何透过新唐 ISP Tool 更新程序代码

1. 开启 NuMicro_ISP_Programming_Tool，Connection interface 选择 USB。
2. 连接开发版 HSUSB port 及 PC 并将 PC.13 短路到地。
3. 按下 Connect 后按下开发板上的 Reset 按钮，确认连上 tool。
4. 点选 APROM 并载入刚刚加密完成的更新文件，勾选 Programing 区域 APROM 后按下 Start。



5. 将 PC.13 改连接到 V_{cc}，当程序侦测到输入信号由低电平拉到高电平后，会自动跳转到 APROM 执行，可在串口调试助手上看到程序执行画面。



6. 修订纪录

Date	Revision	Description
2024.3.29	1.00	初始发布。

Important Notice

Nuvoton Products are neither intended nor warranted for usage in systems or equipment, any malfunction or failure of which may cause loss of human life, bodily injury or severe property damage. Such applications are deemed, "Insecure Usage".

Insecure usage includes, but is not limited to: equipment for surgical implementation, atomic energy control instruments, airplane or spaceship instruments, the control or operation of dynamic, brake or safety systems designed for vehicular use, traffic signal instruments, all types of safety devices, and other applications intended to support or sustain life.

All Insecure Usage shall be made at customer's risk, and in the event that third parties lay claims to Nuvoton as a result of customer's Insecure Usage, customer shall indemnify the damages and liabilities thus incurred by Nuvoton.

Please note that all data and specifications are subject to change without notice.
All the trademarks of products and companies mentioned in this datasheet belong to their respective owners.